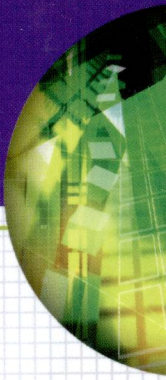


**ELECTRONIC  
ENGINEERING**

# **HARDWARE IMPLEMENTATION of FINITE-FIELD ARITHMETIC**



**JEAN-PIERRE DESCHAMPS  
JOSÉ LUIS IMAÑA  
GUSTAVO D. SUTTER**

TP360.21  
D446

# Hardware Implementation of Finite-Field Arithmetic

Jean-Pierre Deschamps  
José Luis Imaña  
Gustavo D. Sutter



E2010000991

**Mc  
Graw  
Hill**

New York Chicago San Francisco  
Lisbon London Madrid Mexico City  
Milan New Delhi San Juan  
Seoul Singapore Sydney Toronto

Cataloging-in-Publication Data is on file with the Library of Congress.

McGraw-Hill books are available at special quantity discounts to use as premiums and sales promotions, or for use in corporate training programs. To contact a special sales representative, please visit the Contact Us page at [www.mhprofessional.com](http://www.mhprofessional.com).

### **Hardware Implementation of Finite-Field Arithmetic**

Copyright © 2009 by The McGraw-Hill Companies, Inc. All rights reserved. Printed in the United States of America. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a data base or retrieval system, without the prior written permission of the publisher.

1 2 3 4 5 6 7 8 9 0 DOC/DOC 0 1 4 3 2 1 0 9

ISBN 978-0-07-154581-5

MHID 0-07-154581-6

The pages within this book were printed on acid-free paper.

**Sponsoring Editor**

Stephen S. Chapman

**Acquisitions Coordinator**

Alexis Richard

**Editorial Supervisor**

David E. Fogarty

**Project Manager**

Somya Rustagi

**Copy Editor**

Megha Roy Chaudhary

**Proofreader**

Barbara Danziger

**Indexer**

Jean-Pierre Deschamps

**Production Supervisor**

Richard C. Ruzycka

**Composition**

International Typesetting  
and Composition

**Art Director, Cover**

Jeff Weeks

Information contained in this work has been obtained by The McGraw-Hill Companies, Inc. ("McGraw-Hill") from sources believed to be reliable. However, neither McGraw-Hill nor its authors guarantee the accuracy or completeness of any information published herein, and neither McGraw-Hill nor its authors shall be responsible for any errors, omissions, or damages arising out of use of this information. This work is published with the understanding that McGraw-Hill and its authors are supplying information but are not attempting to render engineering or other professional services. If such services are required, the assistance of an appropriate professional should be sought.



# Hardware Implementation of Finite-Field Arithmetic

### About the Authors

**Jean-Pierre Deschamps** received an MS degree in electrical engineering from the University of Louvain, Belgium, in 1967, a PhD degree in computer science from the Autonomous University of Barcelona, Spain, in 1983, and a PhD degree in electrical engineering from the Polytechnic School of Lausanne, Switzerland, in 1984. He worked in several companies and universities. He is currently a professor at the University Rovira i Virgili, Tarragona, Spain. His research interests include ASIC and FPGA design, digital arithmetic, and cryptography. He is the author of seven books and about a hundred international papers.

**José Luis Imaña** received the MS and PhD degrees, both in Physics, from Complutense University of Madrid, Spain, where he is currently a professor. His research interests include algorithms and VLSI architectures for computations in finite fields, cryptography, computer arithmetic, reconfigurable computing architectures, and formal methods in verification. He is the author of about 30 international papers and communications.

**Gustavo D. Sutter** received an MS degree in computer science from State University UNCPBA of Tandil (Buenos Aires), Argentina, and a PhD degree from the Autonomous University of Madrid, Spain. He has been a professor at the UNCPBA, Argentina, and is currently a professor at the Autonomous University of Madrid, Spain. His research interests includes ASIC and FPGA design, digital arithmetic, and development of embedded systems. He is the author of one book and about 30 international papers and communications.

---

# Preface

**F**inite fields are used in different types of computers and digital communication systems. Two well-known examples are error-correction codes and cryptography. The traditional way of implementing the corresponding algorithms is software, running on general-purpose processors or on digital-signal processors. Nevertheless, in some cases the time constraints cannot be met with instruction-set processors, and specific hardware must be considered, that is, circuits specifically designed for executing those complex algorithms: they implement the particular computation primitives of the algorithms and profit from their inherent parallelism.

Apart from the application-specific integrated circuits (ASICs) solution, another technology at hand for developing specific circuits is constituted by field-programmable gate arrays (FPGA). They form an attractive option for small production quantities as their nonrecurrent engineering costs are much lower than those corresponding to ASICs. They also offer flexibility and fast time-to-market. Furthermore, in order to reduce their size, and so the unit cost, an interesting possibility is to reconfigure them at run time so that the same programmable device can execute different predefined functions.

This book describes algorithms and circuits for executing the main finite-field operations, that is, addition, subtraction, multiplication, squaring, exponentiation, and division. It is mainly addressed to hardware engineers involved in the development of embedded systems, including finite-field operations. Distinguishing features of this book are the following:

- The emphasis is different from the classic texts on finite fields. It is not limited to the description of algebraic and algorithmic aspects. The main topic is circuit synthesis.
- A special importance has been given to FPGA implementations. The particular architecture of these components leads the designer to use synthesis techniques somewhat different than the ones applied for ASIC for which standard cell libraries exist. Throughout the book examples of FPGA implementation are described.

- Most algorithms are described in Ada, a programming language similar to VHDL, so that they can be executed and the correctness of the proposed algorithms can be verified with actual input data.
- In what concerns the description of the circuits, logic schemes are presented as well as VHDL models, in such a way that the corresponding circuits can be easily simulated and synthesized.

---

## Overview

The book is divided into 10 chapters. The first chapter (mathematical background) gives the main definitions and properties of finite fields. Chapters 2 to 4 are dedicated to the operations modulo  $m$  and the corresponding circuits. Chapter 2 deals with the modulo  $m$  reduction, Chap. 3 with the modulo  $m$  addition, subtraction, multiplication, and exponentiation, and Chap. 4 with the modulo  $p$  division, where  $p$  is a prime. Chapters 5 and 6 are dedicated to the operations modulo  $f(x)$ , where  $f(x)$  is a polynomial over a finite field, and to the corresponding circuits. Chapter 5 deals with the modulo  $f(x)$  addition, subtraction, multiplication, and exponentiation, and Chap. 6 with the modulo  $f(x)$  division, where  $f(x)$  is an irreducible polynomial. Chapters 7 to 9 are dedicated to the main arithmetic operations over  $GF(2^m)$ . In Chap. 7 polynomial bases are considered (thus, a particular case of the topics dealt with in Chaps. 5 and 6). In Chap. 8 normal bases are used, and in Chap. 9 dual and triangular bases are considered. Chapter 10 is dedicated to elliptic-curve cryptography, currently one of the main finite-field applications.

There are four appendices. Three of them describe circuits for performing arithmetic operations over some particular fields, namely a prime field  $GF(2^{192} - 2^{64} - 1)$  in App. A, two optimal extension fields  $GF(2^{39})$  and  $GF((2^{32} - 387)^6)$  in App. B, and two binary extension fields  $GF(2^{163})$  and  $GF(2^{233})$  in App. C. Appendix D is a brief comparison of the syntaxes of Ada and VHDL.

All the chapters, but the first one, include algorithms, circuits, and results of FPGA implementations. The algorithms are described in Ada and the circuits are modeled in VHDL. Complete and executable source files (Ada and VHDL) are available at the authors' Web site [www.arithmetic-circuits.org](http://www.arithmetic-circuits.org).

---

# Acknowledgments

**T**he authors are grateful to the following universities for providing them the means for carrying this work through to a successful conclusion: University Rovira i Virgili (Tarragona, Spain), Autonomous University of Madrid (Spain), and Complutense University of Madrid (Spain).



# Contents

Preface .....	xi
Acknowledgments .....	xiii
<b>1 Mathematical Background .....</b>	<b>1</b>
1.1 Number Theory .....	1
1.1.1 Basic Definitions .....	1
1.1.2 Euclidean Algorithms .....	2
1.1.3 Congruences .....	4
1.2 Algebra .....	8
1.2.1 Groups .....	8
1.2.2 Rings .....	9
1.2.3 Fields .....	10
1.2.4 Polynomials .....	11
1.2.5 Congruences of Polynomials .....	15
1.3 Finite Fields .....	17
1.3.1 Basic Properties .....	17
1.3.2 Field Extensions .....	18
1.3.3 Roots of Irreducible Polynomials ...	20
1.3.4 Bases of Finite Fields .....	20
1.3.5 Finite Fields $GF(2^m)$ .....	22
1.4 References .....	23
<b>2 mod <math>m</math> Reduction .....</b>	<b>25</b>
2.1 Integer Division .....	25
2.1.1 Digit Recurrence Algorithms .....	25
2.1.2 Nonrestoring Reducer .....	27
2.1.3 SRT Reducer .....	29
2.2 Reduction mod $2^k - a$ .....	33
2.3 Precomputation of $2^{ik}$ mod $m$ .....	38
2.4 Barrett Reduction Algorithm .....	43
2.4.1 $n$ -Digit to $(k + t)$ -Digit Reduction ...	43
2.4.2 An Approximation of $q$ .....	44
2.5 Comparison .....	48
2.6 Specific Circuits .....	49
2.6.1 mod 239 Reducer .....	49
2.6.2 mod $(2^{192} - 2^{64} - 1)$ Reducer .....	50
2.7 FPGA Implementation .....	54
2.7.1 Nonrestoring Reducers .....	55
2.7.2 SRT Reducers .....	55
2.7.3 Reduction mod $2^k - a$ .....	55

2.7.4	Precomputation of $2^{ik} \bmod m$	57
2.7.5	Barrett Reduction	58
2.7.6	Specific Circuits	59
2.8	Comments and Conclusions	59
2.9	References	60
<b>3</b>	<b>mod <math>m</math> Operations</b>	<b>61</b>
3.1	Addition mod $m$	61
3.2	Subtraction mod $m$	63
3.3	Adder/Subtractor mod $m$	64
3.4	Multiplication mod $m$	66
3.4.1	Multiply and Reduce	66
3.4.2	Double, Add, and Reduce	70
3.4.3	Montgomery Multiplication	75
3.4.4	Comparison	81
3.5	Exponentiation	82
3.6	FPGA Implementations	87
3.6.1	mod $m$ Adders/Subtractors	87
3.6.2	mod $m$ Multipliers	87
3.6.3	mod $m$ Exponentiators	88
3.7	Comments and Conclusions	88
3.8	References	89
<b>4</b>	<b>Operations over <math>GF(p)</math></b>	<b>91</b>
4.1	Euclidean Algorithm	92
4.1.1	Integer Division	93
4.1.2	Multiplication and Subtraction	96
4.1.3	mod $p$ Division	98
4.2	Binary Algorithm	100
4.3	Plus-Minus Algorithm	104
4.4	Fermat's Little Theorem	110
4.5	Comparison	112
4.6	FPGA Implementations	113
4.6.1	Euclidean Algorithm	113
4.6.2	Binary Algorithm	114
4.6.3	Plus-Minus Algorithm	114
4.6.4	Fermat's Little Theorem	115
4.7	Comments and Conclusions	116
4.8	References	116
<b>5</b>	<b>Operations over <math>Z_p[x]/f(x)</math></b>	<b>117</b>
5.1	Addition and Subtraction mod $f(x)$	117
5.2	Multiplication mod $f(x)$	121
5.2.1	Two-Step Multiplication	121
5.2.2	Serial Multiplication	123
5.3	Exponentiation mod $f(x)$	128

5.4	Optimal Extension Fields .....	132
5.5	FPGA Implementations .....	136
5.5.1	Adders of Polynomials mod $p$ .....	136
5.5.2	Subtractors of Polynomials mod $p$ .....	136
5.5.3	Adders/Subtractors of Polynomials mod $p$ .....	137
5.5.4	Serial Multipliers .....	137
5.5.5	Exponentiation .....	137
5.6	Comments and Conclusions .....	138
5.7	References .....	138
<b>6</b>	<b>Operations over <math>GF(p^m)</math> .....</b>	<b>139</b>
6.1	Euclidean Algorithm .....	140
6.2	Binary Algorithm .....	147
6.3	Reduction to Multiplications over $GF(p^m)$ and Inversion over $Z_p$ .....	154
6.4	Optimal Extension Fields .....	156
6.5	FPGA Implementations .....	162
6.6	Comments and Conclusions .....	162
6.7	References .....	162
<b>7</b>	<b>Operations over <math>GF(2^m)</math>—Polynomial Bases .....</b>	<b>163</b>
7.1	Multiplication .....	164
7.1.1	Two-Step Classic Multiplication ....	164
7.1.2	Karatsuba-Ofman Polynomial Multiplication .....	169
7.1.3	Interleaved Multiplication .....	171
7.1.4	Matrix-Vector Multipliers .....	174
7.1.5	Montgomery Multiplication .....	182
7.2	Squaring .....	187
7.3	Exponentiation .....	195
7.4	Division .....	204
7.5	Inversion .....	206
7.6	Important Irreducible Polynomials .....	213
7.6.1	Equally Spaced Polynomials (ESPs) .....	213
7.6.2	General Irreducible Polynomials ...	214
7.6.3	All-One Polynomials (AOPs) .....	216
7.6.4	Trinomials .....	219
7.6.5	Pentanomials .....	221
7.7	FPGA Implementations .....	223
7.7.1	Classic Multipliers .....	224
7.7.2	Interleaved Multiplication .....	224
7.7.3	Mastrovito Multipliers .....	224

7.7.4	Mastrovito Multipliers, Second Version .....	225
7.7.5	Interleaved Multiplication, Advanced Version .....	225
7.7.6	Montgomery Multipliers .....	225
7.7.7	Classic Squaring .....	227
7.7.8	LSB First Squarer, Second Version .....	227
7.7.9	Montgomery Squarer .....	228
7.7.10	Binary Exponentiation .....	228
7.7.11	Montgomery Exponentiation .....	229
7.7.12	Division .....	229
7.7.13	Extended Euclidean Algorithm (EEA) for Inversion .....	229
7.7.14	Modified Almost Inverse Algorithm (MAIA) for Inversion .....	230
7.7.15	Important Irreducible Polynomials .....	230
7.8	Comments and Conclusions .....	231
7.9	References .....	231
8	<b>Operations over <math>GF(2^m)</math>—Normal Bases</b> .....	<b>235</b>
8.1	Some Properties of Normal Bases .....	236
8.2	Squaring .....	238
8.3	Multiplication .....	238
8.4	Exponentiation .....	249
8.5	Inversion .....	255
8.6	Optimal Normal Bases .....	259
8.7	FPGA Implementations .....	264
8.7.1	Multiplier .....	265
8.7.2	Exponentiation .....	265
8.7.3	Inversion .....	266
8.7.4	Type-I Optimal Normal Basis Multiplier with AOPs .....	266
8.8	Comments and Conclusions .....	266
8.9	References .....	267
9	<b>Operations over <math>GF(2^m)</math>—Other Bases</b> .....	<b>269</b>
9.1	Dual Bases .....	269
9.2	Triangular Bases .....	277
9.3	References .....	284
10	<b>An Example of Application—Elliptic Curve Cryptography</b> .....	<b>287</b>
10.1	Public-Key Cryptography .....	287
10.2	Elliptic Curve over a Finite Field .....	288

10.3	Group Law	290
10.4	Point Multiplication	292
10.4.1	Definition	292
10.4.2	Basic Algorithms	293
10.4.3	Some Alternative Methods	294
10.5	Example of Implementation	304
10.5.1	Computation Resources	305
10.5.2	Point Addition	305
10.5.3	Point Multiplication	306
10.6	FPGA Implementation	310
10.7	References	311
<b>A</b>	<b><math>p = 2^{192} - 2^{64} - 1</math></b>	<b>313</b>
A.1	Hexadecimal Representation	313
A.2	mod $p$ Reduction	313
A.2.1	Generic Sequential Circuit	313
A.2.2	Specific Combinational Circuit	314
A.2.3	FPGA Implementation	314
A.3	mod $p$ Addition and Subtraction	314
A.4	mod $p$ Multiplication	315
A.4.1	Generic Circuit	315
A.4.2	Specific Circuit	315
A.5	mod $p$ Exponentiation	316
A.6	mod $p$ Division	317
<b>B</b>	<b>Optimal Extension Fields</b>	<b>319</b>
B.1	$GF(2^{39^{17}})$	319
B.1.1	VHDL Models and Constant Definitions	319
B.1.2	FPGA Implementations	320
B.2	$GF((2^{32} - 387)^6)$	321
B.2.1	Constants	321
B.2.2	mod $p$ Reduction	323
B.2.3	mod $p$ Addition and Subtraction	323
B.2.4	mod $p$ Multiplication	324
B.2.5	mod $p$ Division	324
B.2.6	mod $(x^6 - 2)$ Multiplication	325
B.2.7	mod $(x^6 - 2)$ Division	326
<b>C</b>	<b>Binary Fields</b>	<b>331</b>
C.1	$GF(2^{163})$	331
C.1.1	mod $f(x)$ Multiplication	331
C.1.2	mod $f(x)$ Division	331
C.1.3	Squaring	332
C.1.4	Elliptic-Curve Operations	332



C.2	$GF(2^{233})$ .....	333
C.2.1	mod $f(x)$ Multiplication .....	333
C.2.2	mod $f(x)$ Division .....	334
C.2.3	Squaring .....	334
C.2.4	Elliptic-Curve Operations .....	334
D	Ada versus VHDL .....	337
	Index .....	341

# CHAPTER 1

---

## Mathematical Background

This chapter presents some topics in mathematics; it is intended to make this book self-contained. For further details the reader can refer to textbooks on Algebra ([Coh93], [GN03], [Her75], [Hun74]), Number Theory ([Kob94], [Ros92], [Ros00], [Gar59]), Finite Fields ([LN83], [LN94], [McC87], [Men93]), and Cryptography [MOV96], from where the following material has been mainly extracted.

---

### 1.1 Number Theory

#### 1.1.1 Basic Definitions

##### Definitions 1.1

1. The set of natural numbers<sup>1</sup>  $N = \{0, 1, 2, 3, \dots\}$ .
2. The set of integers  $Z = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$ .

**Definition 1.2** Given two integers  $x$  and  $y$ ,  $y$  divides  $x$  ( $y$  is a *divisor* of  $x$ ) if there exists an integer  $z$  such that  $x = zy$ .

**Definition 1.3** Given two integers  $x$  and  $y$ , with  $y > 0$ , there exist two integers  $q$  (the *quotient*) and  $r$  (the *remainder*) such that

$$x = qy + r \quad \text{where } 0 \leq r < y$$

It can be proven that  $q$  and  $r$  are unique.

Then (notation)

$$r = x \bmod y \quad q = x \operatorname{div} y$$

An alternative definition:

---

<sup>1</sup>For convenience, the element zero has been included in  $N$ .

**Definition 1.4** (integer division) Given two integers  $x$  and  $y$ , with  $y > 0$ , there exist two integers  $q$  (the *quotient*) and  $r$  (the *remainder*) such that

$$x = qy + r \quad \text{where } 0 \leq r < y \text{ if } x \geq 0 \text{ and } -y < r \leq 0 \text{ if } x < 0$$

It can be proven that  $q$  and  $r$  are unique. Then (notation)

$$r = x \text{ rem } y \quad q = x/y$$

### Examples 1.1

1.  $x = -16, y = 3$ :

$$-16 \bmod 3 = 2, -16 \operatorname{div} 3 = -6, -16 = -6 \cdot 3 + 2$$

$$-16 \text{ rem } 3 = -1, -16/3 = -5, -16 = -5 \cdot 3 + (-1)$$

2.  $x = -15, y = 3$ :

$$-15 \bmod 3 = 0, -15 \operatorname{div} 3 = -5, -15 = -5 \cdot 3 + 0$$

$$-15 \text{ rem } 3 = 0, -15/3 = -5, -15 = -5 \cdot 3 + 0$$

### Definitions 1.5

- Given two integers  $x$  and  $y$ ,  $z$  is the *greatest common divisor* of  $x$  and  $y$  if
  - $z$  is a natural number (nonnegative integer),
  - $z$  divides both  $x$  and  $y$ ,
  - any other common divisor of  $x$  and  $y$  is also a divisor of  $z$ .

Notation:  $z = \gcd(x, y)$ .

- Given two integers  $x$  and  $y$ , they are said to be *relatively prime* if  $\gcd(x, y) = 1$ .
- An integer  $p > 1$  is said to be *prime* if its only positive divisors are 1 and  $p$ .

## 1.1.2 Euclidean Algorithms

Given two natural numbers  $x$  and  $y$ , the *Euclidean algorithm* for natural numbers computes  $\gcd(x, y)$ . It is based on a series of integer divisions:

$$r(i-1) = q(i)r(i) + r(i+1) \quad \text{where } 0 \leq r(i+1) < r(i)$$

Observe that any divider of  $r(i-1)$  and  $r(i)$  is also a divider of  $r(i)$  and  $r(i+1)$  so that

$$\gcd(r(i-1), r(i)) = \gcd(r(i), r(i+1))$$

Initially

$$r(0) = x \quad \text{and} \quad r(1) = y$$

Then compute

$$r(0) = q(1)r(1) + r(2)$$

$$r(1) = q(2)r(2) + r(3)$$

$$r(2) = q(3)r(3) + r(4)$$

...

$$r(n-3) = q(n-2)r(n-2) + r(n-1)$$

$$r(n-2) = q(n-1)r(n-1) + r(n)$$

where  $r(1) > r(2) > \dots > r(n) = 0$  and  $\gcd(r(i-1), r(i)) = \gcd(r(i), r(i+1))$ , so that

$$\gcd(x, y) = \gcd(r(0), r(1)) = \dots = \gcd(r(n-1), r(n)) = \gcd(r(n-1), 0) = r(n-1)$$

**Example 1.2** Let  $r(0) = x = 9520$ ;  $r(1) = y = 3120$ ;

$$9520 = 3 \cdot 3120 + 160$$

$$3120 = 19 \cdot 160 + 80$$

$$160 = 2 \cdot 80 + 0$$

Then  $\gcd(9520, 3120) = 80$ .

In the *extended Euclidean algorithm* a series of coefficients  $b(i)$  and  $c(i)$  is calculated in parallel with the computation of  $r(0)$ ,  $r(1)$ ,  $r(2)$ ,  $\dots$ ,  $r(n)$ :

$$b(0) = 1$$

$$c(0) = 0$$

$$b(1) = 0$$

$$c(1) = 1$$

$$b(2) = b(0) - b(1)q(1)$$

$$c(2) = c(0) - c(1)q(1)$$

...

$$b(n-1) = b(n-3) - b(n-2)q(n-2)$$

$$c(n-1) = c(n-3) - c(n-2)q(n-2)$$

It can be demonstrated by induction that

$$r(i) = b(i)x + c(i)y \quad \forall i = 0, 1, 2, \dots, n-1$$

In particular

$$\gcd(x, y) = r(n-1) = b(n-1)x + c(n-1)y$$

In conclusion the extended Euclidean algorithm expresses the greatest common divisor  $z$  of two natural numbers  $x$  and  $y$  as a linear combination of  $x$  and  $y$ , that is,

$$z = bx + cy \quad (1.1)$$